An evolutionary scheme for decision tree construction

Nour El Islem Karabadji, Hassina Seridi, Fouad Bousetouane, Wajdi Dhifli, Sabeur Aridhi

An Evolutionary Scheme for Decision Tree Construction

Nour El Islem Karabadji[a,b,*], Hassina Seridi[b], Fouad Bousetouane[c], Wajdi Dhifli[d], Sabeur Aridhi[e]

[a] *Preparatory School of Science and Technology, Annaba, P.O. Box 218, 23000, Algeria.*
[b] *Electronic Document Management Laboratory (LabGED), Badji Mokhtar-Annaba University, P.O. Box 12 Annaba, Algeria.*
[c] *Real Time Intelligent System Laboratory, University of Nevada, Las Vegas, NV 89154, USA.*
[d] *Department of Computer Science, University of Quebec At Montreal, PO box 8888, Downtown station, Montreal (Quebec) Canada, H3C 3P8.*
[e] *University of Lorraine, LORIA, Campus Scientifique, BP 239, 54506 Vandoeuvre-lès-Nancy, France.*

Abstract

Classification is a central task in machine learning and data mining. Decision tree (DT) is one of the most popular learning models in data mining. The performance of a DT in a complex decision problem depends on the efficiency of its construction. However, obtaining the optimal DT is not a straightforward process. In this paper, we propose a new evolutionary meta-heuristic optimization based approach for identifying the best settings during the construction of a DT. We designed a genetic algorithm coupled with a multi-task objective function to pull out the optimal DT with the best parameters. This objective function is based on three main factors: (1) *Precision* over the test samples, (2) *Trust* in the construction and validation of a DT using the smallest possible training set and the largest possible testing set, and (3) *Simplicity* in terms of the size of the generated candidate DT, and the used set of attributes. We extensively evaluate our approach on 13 benchmark datasets and a fault diagnosis dataset. The results show that

*Corresponding author. Address: LabGED, Badji Mokhtar University, PO Box 12, 23000, Annaba, Algéria. Tel: 0021338872678; Fax: 0021338872436

Email addresses: karabadji@labged.net (Nour El Islem Karabadji), seridi@labged.net (Hassina Seridi), fouad.bousetouane@unlv.edu (Fouad Bousetouane), dhifli.wajdi@courrier.uqam.ca (Wajdi Dhifli), sabeur.aridhi@loria.fr (Sabeur Aridhi)

it outperforms classical DT construction methods in terms of accuracy and simplicity. They also show that the proposed approach outperforms Ant-Tree-Miner (an evolutionary DT construction approach), Naive Bayes and Support Vector Machine in terms of accuracy and F-measure.

Keywords: Decision Trees, Genetic Algorithms, Attributes Selection, Data Reduction.

1. Introduction

Data Mining is the process of extracting information and knowledge from huge datasets. Classification is one of the most popular tasks in data mining. It involves building a classification model (classifier) based on a training set of labeled examples, where the generated classifier should be able to classify new instances properly. In general, the training examples are labeled by an expert or based on some experimental results (*e.g.*, weather observations). The classifier is supposed to discover significant and hidden patterns over the training data that allow it to predict the labels/classes of new instances efficiently. Technically, the classification model is parameterized according to the characteristics of the training examples. However, several models could be estimated for the same training set. Obtaining the model with the highest precision and generalization capabilities is not trivial. Indeed, it relies on finding the optimal model $h^*(x)$ that best minimizes the loss (cross entropy loss, hinge loss, *etc.*) between the predicted labels and the ground truth for examples of an evaluation set.

In the literature, a wide spectrum of different classification approaches was developed such as support vector machines [16, 24], bayesian inference based approaches [1, 33], neural networks [26, 34] and deep learning [32, 44]. In contrast of these popular approaches, decision trees (DTs) are still preferred for a wide range of data mining problems. This is due to the fact that they encode explicitly semantic relationships between attributes which allow users to understand easily the behavior of the generated models.

The construction of a DT is mainly based on two stages: (1) a growing phase and (2) a pruning phase. The growing phase is the process of splitting the training data repeatedly into two or more subsets in a hierarchical manner. Different algorithms could be used in this stage which involves different growing strategies such as Gini and χ^2. The growing process stops once stopping criteria are satisfied or all instances of each subset wrap the

same class. In general, the growing stage outputs very large-depth DTs with high complexity.

In the pruning phase, different heuristic strategies are often used to reduce the size and complexity of the constructed DT [9, 38, 42, 43]. In fact, the pruning also allows to prevent over-fitting by removing all sections of the DT that might be affected by noisy or imprecise data. However, finding the best trade-off between the pruning level and the prediction accuracy of the DT over the test samples is not an easy task. Since an over-pruning makes a distortion of the classification model and the latter will only represent a small portion of the training set. This engenders an over-generalization of the classification model and the latter will accept many false positives in the prediction. An under-pruning makes the classification model overfits the training data. Generating an accurate and optimized DT for a complex dataset (*e.g.*, non-linearly separable) requires answering the following questions:

- How to choose the most appropriate training samples?
- What are the model parameters that fit better the learning data?
- What are the set of attributes to be considered?
- When to stop the growing phase?
- When to stop the pruning phase?

Choosing a pruning technique for a specific decision problem (*e.g.*, ID3 [30]) is not straightforward and there is no formal justification about the choice of the DT construction technique. If all those questions have to be simultaneously answered, a very complex combinatorial problem must be solved, which evolves exponentially in terms of memory consumption and computation time. Meta-heuristic strategies are known as powerful search algorithms (Genetic Algorithm (GA), Particle Swarm Optimization, *etc.*) for finding the optimal solution for complex combinatorial problems.

In this paper, we present a new powerful approach for optimal DT construction with high accuracy for complex classification/prediction problems. This approach is mainly based on a genetic algorithm that allows finding the optimal DT by taking into account several combinations of the possible parameters, subsets of attributes, and subsets of training/testing examples. Experimental results show that our approach outperforms ordinary

DT construction approaches in terms of prediction accuracy and complexity of the constructed DT model. These results also illustrate a competitive behaviour of our approach with a large magnitude in term of accuracy and F-measure compared to known well established classifiers namely Naive Bayes and Support Vector Machine. In addition, the accuracy and runtime results of the proposed approach have shown good performances compared to an ant colony optimisation algorithm denoted Ant-Tree-Miner [39] that constructs improved DTs. To summarize, the identified contributions of this work are as follow:

- We propose a GA scheme to deal with a range of key strategic choices to improve the accuracy and the simplicity of the DT. To this end, the best chromosome that allows the construction of an optimal DT is selected.

- We design a multi-task fitness function that allows to pull out the most appropriate chromosome according to a set of multiple constraints. These constraints consist of (1) Favoring the improvement of the accuracy of the DT. (2) Reducing the size of the used sets of attributes and training samples, as well as the size of the constructed DT. (3) Increasing the size of the set of testing samples.

- We experimentally evaluate the performance of our approach on various benchmark datasets and we show that it outperforms multiple ordinary DT construction approaches, Naive Bayes and SVM classifiers, and Ant-Tree-Miner.

Our approach can be used in a variety of real-world classification problems where the choice of the best combination of the possible parameters, subsets of attributes, and subsets of training/testing examples is hard. For instance, in protein sequence and structure annotation, classification models are one of most useful techniques to predict the annotation of un-annotated proteins [18, 2]. Mainly, the task is to predict the structural or functional class for each protein based on a set of discriminative patterns (*e.g.* subsequences or substructures) used as attributes and with respect to a reference dataset of known labeled proteins. The problem is that the size of the reference database in real-world cases is usually extremely large (for instance Uniprot [15] counts more than 80 million sequences) making straightforward application of exact classification algorithms extremely costly or even unfeasible. Besides the number of patterns used as attributes could be very high

making the classification task even harder due to the curse of dimensionality [19]. Our approach is highly useful in such cases as it allows to build a simple and efficient DT model by pulling out automatically the best subsets of reference training and test samples as well as the best subset of attributes and parameters. Another domain of application of our approach is recommendation systems which are among the most important tools in many real-world applications such as social networks like Facebook, commercial websites like Amazon, streaming media and video websites like Netflix, *etc.* In this domain of application the system is interested in recommending for each user novel profiles, items or products that could interest him and thus increase the business income. The recommendation is usually based on the similarity between the user of interest and a subset of the most similar users selected from a reference database [6]. The bottleneck here is that the size of the reference database is usually huge thus making the search extremely costly and even sometimes unfeasible. Besides, the number of items and ratings at each profile could also be very large especially for old users. Fast approximation of a decision model that relies on an efficient sampling of a representative set of users and items is highly useful. Our approach could be used to resolve such recommendation problems when the decision could be seen as a binary classification.

The rest of the paper is organized as follows. In Section 2, some previous studies on DT construction are briefly discussed. In Section 3, we describe our evolutionary approach for DT construction. Section 4 reports the experimental results on 13 benchmark datasets as well as a real-world application example of our approach on the problem of fault diagnosis in rotating machines.

2. Related works

DTs are among the most popular methods in data mining. Both the simplicity and efficiency of DTs motivates their wide usage in several research areas including classifier aggregation [8], boosting [20], clustering [23], text mining [48] ,network anomaly and intrusion detection [17, 46], and recommendation systems [14].

We can survey some pertinent works on optimized decision tree construction in [3, 11, 28, 30, 37, 39, 40]. In [39], the authors propose Ant-Tree-Miner, an ant colony optimization (ACO) algorithm to induce decision trees. Ant-Tree-Miner combines commonly used strategies from both traditional deci-

sion tree induction algorithms and ACO. The algorithm starts by initializing the pheromone values and computing the heuristic information for each attribute of the training set. Each ant in the colony creates a new decision tree until a maximum number of iterations is reached or the algorithm has converged. When a tree is created, it is pruned in order to simplify it and thus potentially avoid over-fitting of the model to the training data. After the pruning, the tree is evaluated and compared to the previously constructed trees.

In [40], an algorithm for constructing multivariate decision trees with monotonicity constraints (MMT) is proposed. The splitting hyper-plane of the proposed algorithm is oblique and the proposed algorithm generates the best split by using rank mutual information (RMI) and rank Gini impurity (RGI). In order to solve incomparable issues in monotone classification tasks, a linear discriminant function is introduced. This linear discrimination function is generated using the non-negative least-squares method [49].

In [3], the authors proposed a hyper-heuristic evolutionary algorithm called HEAD-DT that evolves design components of top-down decision-tree induction algorithms. The main goal of HEAD-DT is to generate the best decision-tree algorithm for a given application domain. HEAD-DT can be seen as a regular generational evolutionary algorithm in which individuals are collections of building blocks of top-down decision-tree induction algorithms. The proposed algorithm has been applied to microarray gene expression data sets. Each individual in HEAD-DT is encoded as an integer vector, and each gene can take a value in a predefined range of values. The set of genes is divided into four categories: (1) split genes, (2) stopping criteria genes, (3) missing value genes, and (4) pruning genes.

An attention was also devoted to the optimization of node splitting measures for decision tree construction. In [11], the authors proposed a new node splitting measure that possesses the convexity and cumulative property, which are important properties for any split measure. Similarly to existing splitting measures, the proposed one is designed to reduce the number of distinct classes that would result in each sub-tree after a split.

Moreover, several DT construction problems were addressed in the literature. In [30], the problem of choosing the best model has been addressed where the authors proposed a GA-based approach to pull up the best decision model for a fan fault diagnosis problem. The optimal DT model was chosen using a fixed number of testing samples. However, such a setting may lead

to missing some good samples that could allow constructing a better model.

In [28] a genetic scheme is proposed to select the most appropriate model and to construct the best DT. An overlapping between the subsets of training samples is considered to overcome the fixed number of samples. In spite of the high accuracy of the generated DT in this work, a lot of good training samples may be missed. Moreover, this scheme does not pay any attention to the size of training set which is a key part of data reduction. Data reduction essentially involves dimensionality reduction and/or example reduction [41]. This allows reducing: (1) the data complexity which greatly facilitates the learning phase, and (2) the effects related to unwanted phenomena (*e.g.*, noise, missed values, counter-examples). However, different experiments have shown that using data reduction separately for descriptor (attribute) selection is less efficient than using a hybrid model that combines both. Paying more attention to the attribute selection may ensure a good generalization by preserving only the relevant attributes [4, 27, 35].

In [29], IUDT is proposed as a new algorithm for best attribute selection in the training stage using a decision tree-like classifier. By exploring the search space of the whole possible combinations, the optimal combination consists of the data and the attributes that allow to construct an optimal DT. In fact, IUDT requires a reasonable number of attributes and training subsets which consequently reduces the risk of combinatorial explosion.

Other notably interesting works in the context of optimal DT generation using evolutionary optimization algorithms are [10, 12, 22, 31].

In addition to the input quality while building a robust classifier (*i.e.*, DT in our case), the model parameters should be well tuned. These parameters include the pruning activation, the use of a pre-pruning or a post-pruning scheme, the minimum number of samples in each leaf, the allowed maximum depth, *etc.* Hereby, hyper-parameter selection in this context is a very complex task. Therefore, it is necessary to carry out a detailed study of growing and pruning techniques to choose the most appropriate setting for an optimal DT. In other words, the problem is to find an optimal combination that enables to build an accurate and optimal DT. Unfortunately, just listing combinations may be intractable (*i.e.*, too many elements) which makes choosing the best setting using a brute force approach a difficult, greedy and time consuming task. The problem becomes even more complicated by associating an objective function that allows us to choose the optimal combination. Thus, we propose using a GA to avoid exploring all the search space.

3. An evolutionary approach for decision tree construction

In this section, we present our proposed evolutionary approach that aims to generate the most optimal and accurate DT for complex data mining problems. Table 1, summarizes the used notations.

Table 1: Notation

Notation	***Description***
$\mathcal{D}$	The dataset
N	The population size
M	The GA number of iterations
AT	A set of attributes
A	A subset of attributes of AT
a_i	An attribute of A, $i \in [0..\|AT\|[$
IN	The splitting instance number, $IN = 10$
TR	A partition of training samples
TE	A partition of testing samples
n	The number of sub-training samples
m	The number of sub-testing samples
tr_i	Sub-training samples of TR, $i \in [0..n * IN[$
te_j	Sub-testing samples of TE, $j \in [0..m * IN[$
α	A set of sub-training samples
β	A set of sub-testing samples
λ	The Typical DT size for a given model
C_i	The Chromosome i, $i \in [0..N[$
T	DT models considering set
t_i	DT built using C_i
SP	The whole secondary parameters lists
SP_j	The list of secondary parameters for the DT model j , $j \in [0..\|T\|[$
sp_i	The list of secondary parameters, $i \in [0..\|SP\|[$
AC	The classification accuracy

Figure 1 presents an overview of the proposed approach.

3.1. Pre-processing phase

In general, DT construction approaches often split the dataset into training and testing data with different portions and settings. In [21], the authors proposed a splitting of the dataset between 70% and 30% for training and

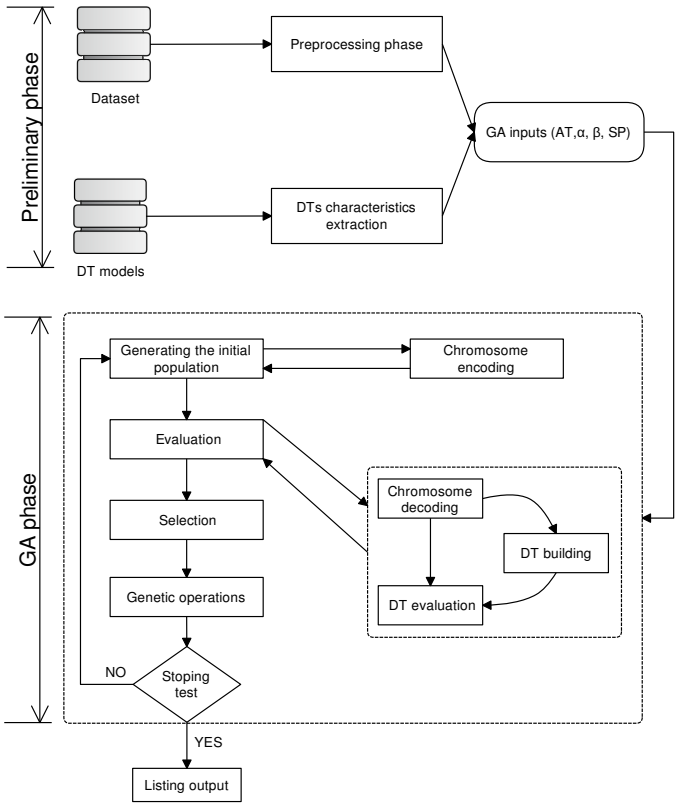

Figure 1: The proposed scheme

test sets respectively. In [36], 60% training and 40% test data are used for the construction of the DT. In our previous work [29], we proved that a special splitting configuration could improve the accuracy of the DT. Multiple instances are used to avoid the assumption that a single random split of

data may give unrepresentative results. In this work, we propose to split the dataset into 2/3 and 1/3 respectively for training (TR) and test (TE) sets, with shuffling and random repetition of 10 times (*i.e.*, $IN = 10$). Moreover, for each couple TR and TE, n and m sample subsets are formed (*i.e.*, $n * tr_i$ and $m * te_j$, $i \in [0..n[$ and $j \in [0..m[$). To get representative results, tr_i and te_j sizes' are set randomly between 30% and 90% of the used TR and TE. This latter choice will guarantee having training samples reduction with at least 10% of the TR size. Figure 2 illustrates the pre-processing procedure.

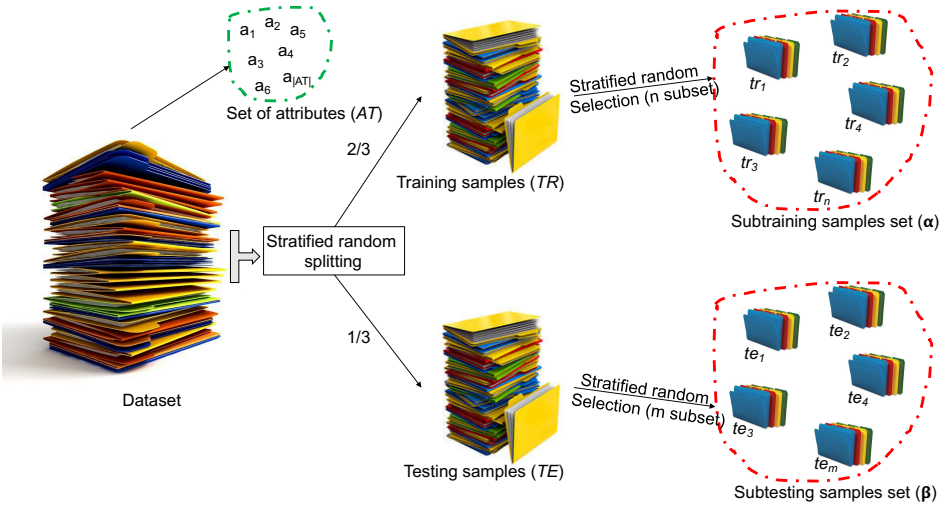

Figure 2: Pre-processing phase

3.2. Meta-heuristic optimization phase

Genetic Algorithms (GAs) are widely used in optimization because of the power of their diversification principle. This principle is based on a random combination between different solutions to reach a global optimum while avoiding a greedy search across the high number of all possible solutions. GAs are based on natural evolution, selection, and survival based on fitness tests [45]. In this work, the population of our GA is a set of 5 different gene vectors.

Each individual represents an instance of the choices list. Calculating a successive population tends to improve a fitness function that aims to improve the DT accuracy and simplicity.

3.2.1. Chromosomical representation

Individuals are represented as an array of 0s and 1s (*i.e.*, binary strings). As we have previously stated, each chromosome is composed of 5 genes. Each gene g_i is represented by X_i bits. Figure 3 illustrates the chromosome representation. g_1 represents the attributes subset A used in the construction process. X_1 bits are used, where X_1 is the size of attributes in the dataset (*i.e.*, $X_1 = |AT|$). g_2 and g_3 respectively represent the identifiers of the sets of sub-training samples tr_i, and sub-testing samples te_j. X_2 and X_3 are the bits required to binary encode the size of the sets α and β. g_4 represents the used DT model. X_4 bits are reserved, where X_4 is defined according to the size of $|T|$ (*i.e.*, the set of considered DT models). g_5 represents the identifier of the list of secondary parameters (sp_i) that is considered in the construction process. g_5 uses X_5 bits which are the minimum of bits required to binary encode the integer that represents the size of the whole set of secondary parameters lists (SP).

Table 2: Example of input instances

Input	***Description***
T	4 models identified by $i \in [0..\|T\|[$ are the identifiers
$\|AT\|$	10 attributes
$\|\mathcal{D}\|$	1000 samples
IN	1 time
n	40 set of sub-training samples
m	20 set of sub-testing samples
$\|SP\|$	10 lists of secondary parameters

Example 1. *Based on the configuration illustrated in the Table 2, the required bits to encode attribute subsets are* $X_1 = |AT| = 10$. *The gene 2 bits' size is* X_2 *bits which allows a binary representation of the size of* $|\alpha|$. *In this case,* $IN * n = 40$ *which needs* 6 *bits* ($X_2 = 6$). *For* $IN * m = 20$ *sub-testing sample sets (i.e.,* $|\beta|$*)* $X_3 = 5$ *bits.* $X_4 = 2$ *bits because 2 bits can binary encode the four models identifiers (i.e.,* $0 \leftrightarrow 00$, $1 \leftrightarrow 01$, $2 \leftrightarrow 10$, *and* $3 \leftrightarrow 11$*).*

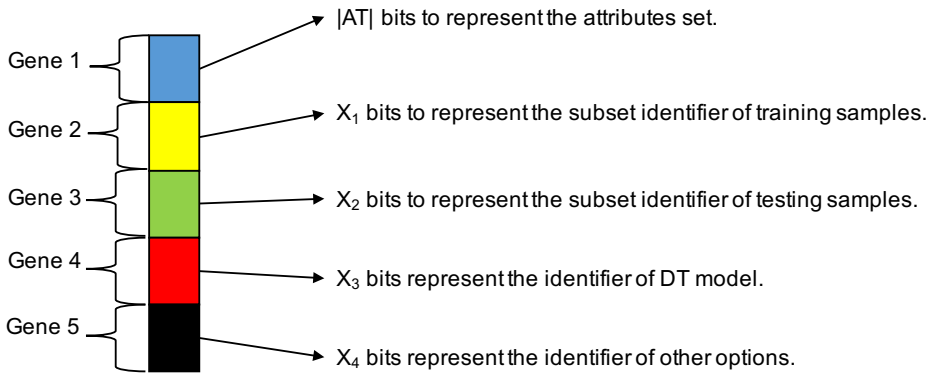

Figure 3: Chromosome representation

$X_5 = 4$ *bits which allows representing the 10 different secondary parameter lists (SP).*

3.2.2. Generation of the initial population

In our approach, the initial population is generated randomly in order to increase the diversity. In fact, a set of N chromosomes is built, Figure 4 illustrates three different chromosomes that are built by considering inputs from the Table 2. The three chromosomes are strings of 27 bits. Genes sizes' are defined as follow: $|g_1| = 10$, $|g_2| = 6$, $|g_3| = 5$, $|g_4| = 2$, and $|g_5| = 4$.

3.2.3. Evaluation and selection of individuals

The selection phase looks for picking some chromosomes of the population for reproduction, which is based on the fitness function. Many selection strategies were proposed in the state-of-the-art [45]. In this work, we use the steady-state selection strategy. The GA is mainly based on two steps:

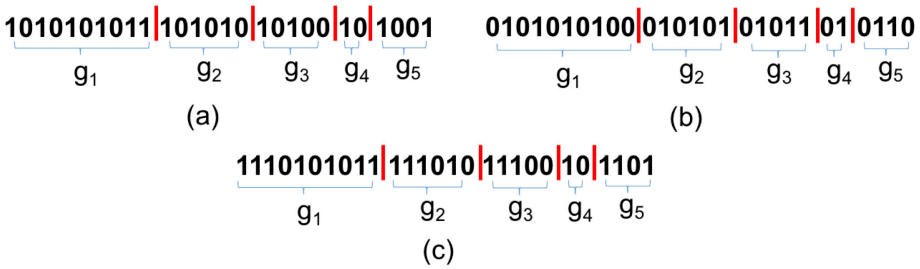

Figure 4: Chromosome encoding instances

1. Chromosome selection: in each generation, half of the population is selected among the best to create new children. Subsequently, the worst chromosomes are removed and replaced with new ones.
2. A fitness/objective function is used to evaluate each solution (chromosome). This function is estimated according to an individual C_i.

3.2.4. Decoding a chromosome:

The decoding process consists in converting the binary gene codes of the encoded chromosome C_i into the appropriate setting. Technically, this conversion generates multiple parameters: 1) The attributes subset A from the gene 1 binary code. 2) The identifiers i, j of the sub-training tr_i and sub-testing te_j samples sets respectively. 3) The DT model identifier. 4) The identifier i of the secondary parameter list sp_i. These parameters are decoded as follows:

- A is composed of attributes a_i for which i in the substring gene 1 are 1s.

- tr_i is the sub-training set with the index i. This one is gotten by converting the substring gene 2 to an integer X, then $i = modulo(X, IN * n)$. For tr_j, j is given as $j = modulo(X, IN * m)$, where X is the result of converting gene 3 to an integer.
- The DT model is identified using the gene 4 (i.e., X is the integer coded in gene 4), where the identifier is the result of $modulo(X, |T|)$.
- Secondary parameters are generated by calculating the integer X coded in gene 5, then the sp_i is identified, where $i = modulo(X, |SP_j|)$ and j is the DT model identifier.

The Example 2 illustrates a decoding example.

Example 2. *Let us assume using Table 2 attributes as the input, the chromosomes presented in Figure 4, and a distribution of the secondary parameters lists as follows: $|SP_0|$: two lists, $|SP_1|$: one list, $|SP_2|$: five lists, and $|SP_3|$: two lists.*

Chromosome (a) corresponds to: the subset of attributes $A = \{a_0, a_2, a_4, a_6, a_8, a_9\}$, the subset of training samples tr_2, the subset of testing samples te_0, the DT model that is identified by 2, and the secondary parameters list sp_2.

Chromosome (b) corresponds to: the subset of attributes $A = \{a_1, a_3, a_5, a_7\}$, the subset of training samples tr_{21}, the subset of testing samples te_{11}, the DT model that is identified by 1, and the secondary parameters list sp_0.

Chromosome (c) corresponds to: the subset of attributes $A = \{a_0, a_1, a_2, a_4, a_6, a_8, a_9\}$, the subset of training samples tr_{18}, the subset of testing samples te_8, the DT model that is identified by 2, and the secondary parameters list sp_3.

3.2.5. Fitness function:

In the proposed evolutionary approach, the fitness function must weight the chromosomes based on their performance in the construction of an optimized and accurate DT. For this end, the selected chromosome is the one that is generated by using a small subset of attributes, a small training subset, and by setting a good secondary parameters list. Moreover, the testing subset must be significant, the DT size must be small, and the accuracy must be as high as possible. The proposed fitness function is defined as:

$$f(x) = 0.66 * Precision + 0.17 * Trust + 0.17 * Simplicity. \quad (1)$$

We can see that the fitness measure is also composed of three factors.

1. Precision is the performance of the DT over a set of testing samples, it is computed as the classification accuracy (AC).
2. Trust is a measure of confidence: we seek to favor chromosomes that lead to construct DTs using small training sets while showing good AC performances on large testing sets.
3. The last factor is the simplicity, which favors the DTs of small sizes and the ones constructed using small attribute subsets.

Formally, the three factors are defined as follows:

$$Precision = AC(DT(tr, A, sp), te) \tag{2}$$

$$Trust = 1 - (Precision/(\frac{(1 - \frac{|tr|*100}{|\overline{TR}|}) + (\frac{|te|*100}{|\overline{TE}|})}{2}) \tag{3}$$

$$Simplicity = (1 - \frac{|A|}{|AT|}) + 2 * \frac{e^{ln(\lambda * size(DT))/e^{size(DT)/\lambda}}}{\lambda^2} \tag{4}$$

Where $|\overline{TR}|$, $|\overline{TE}|$ are the average sizes of training and testing partitions respectively. $|AT|$ is the size of the set of attributes, λ is the size of a reference DT model according to this input dataset. Assuming a chromosome C_i, A is the subset of attributes, tr is the subset of training samples, te is the subset of testing samples, and sp is a secondary parameters list. The evaluated DT model is constructed using tr by considering only the attributes in A, and the secondary parameters list sp, $size(DT)$ is the DT size. Practically, the output of the designed fitness function is bounded in the range of [0,1].

3.2.6. Genetic operators

Crossover. this operator is applied on two different individuals. As a result, it gives a chromosome formed from the fusion of the characteristics of both parents, two children are generated for the next generation. A percentage of crossovers is set to 99%, and a cross at a point is applied.

Mutation. this operation is carried out through the modification of one or more genes, chosen randomly from the parents. In our case, the used ratio of the mutation is set to 1%. This ratio defines the probability of changing a bit by another randomly selected one without interaction with other chromosomes.

3.2.7. Stopping Criterion

The algorithm stops at one of the following criteria:

- the maximum number of iterations is equal to M.
- 50% of the population of chromosomes are similar to the first and the fourth genes.

3.2.8. Complexity analyses

In this section, we study the complexity of the proposed genetic algorithm for constructing an improved decision tree. As revealed earlier, the proposed approach is composed of two main phases: a preliminary phase and a GA phase. However, the major computational cost is used over the second phase. For GA phase, there are three main steps that are repeated M times. These steps are the computation of fitness, the selection process, and the genetic operations. Assuming that δ and σ are the times required to learn and evaluate a given DT model and $|ch|$ is a chromosome size. By considering decoding, learning, and evaluation phases, the fitness computation phase requires $O(N * (|ch| + \delta + \sigma))$ in the worst case. For the selection phase, the problem can be seen like an urn problem for which $N/2$ of the population is drawing. The complexity in the worst case of this latter phase is bounded by $O(N^2)$. The third phase consists of the application of genetic operators for which the complexity is given by $O(N * |ch|)$. Finally, we can give the complexity of the proposed GA used to construct an improved decision tree by $O(N^2 * (N + \delta + \sigma + |ch|))$ in the worst case.

4. Experiments

The proposed evolutionary approach for DT construction is implemented in Java using the WEKA framework [25]. We validated the proposed approach on two scenarios. In the first scenario, we applied our approach on 13 machine learning benchmarks extracted from the UCI collection [5]. In the second scenario, the proposed approach is applied to a real-world application example on fault diagnosis in a rotating machine.

To have a clear idea of the proposed GA performances, first, we compare the experimental results of our approach with those of WEKA's DTs that was built using the default parameters by using pruning (DTP). Second, we

Table 3: The used GA configurations

Configuration	N	M	IN	n	m
$conf_1$	100	100	10	10	10
$conf_2$	100	100	10	20	20
$conf_3$	200	100	10	20	30
$conf_4$	200	100	10	50	50

compare GA results with those of Naive Bayes (NB), Radial Basis Function (RBF) kernel [13] Support Vector Machine denoted $SVM(RBF)$, and Polynomial kernel SVM denoted $SVM(POLY)$. Finally, we compare our results with those of Ant-Tree-Miner [39]. We note that the results are obtained following 10-folds cross-validation (10-CV).

Four DT models were chosen to carry out different experiments, namely BFTree, J48, SimpleCart, and REPTree. In addition to the possibility of using pruning methods within their default WEKA implementation, these models are accumulated into fifty-eight possible choice lists. To validate the effectiveness of the proposed GA approach, we considered four configuration settings that are reported in Table 3 (*i.e.*, the population size, the size of sub-training and sub-testing sets, splitting times, and the maximum number of iterations).

4.1. Experimental data

The used UCI datasets are described in Table 4. The %-Base Error column refers to the percentage of error that is obtained if the most frequent class is always predicted.

Tables 5 and 8 list, for each dataset, the GA selected model, the considered number of attributes, the DT sizes, the classification accuracy, and the F-measure.

In Tables 5 and 8, we notice that the number of attributes that are used to construct the DTs represents only about 40% of the original set. The size of DTs generate based on GA are less than the Standard DTs sizes. These later ones are computed as average DTs sizes constructed using default

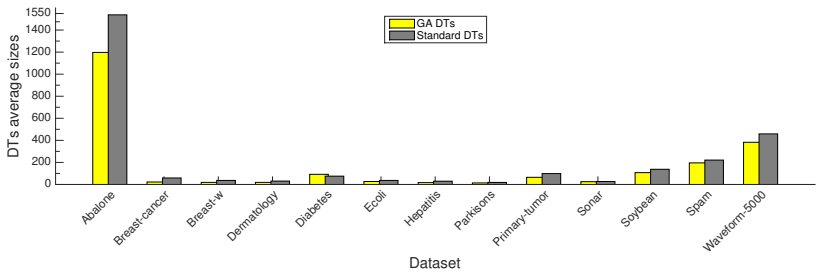

Figure 5: Standard DTs average sizes vs. GA DTs average sizes

parameters. These reductions have a central role in ensuring the construction of a simple yet optimal DT with high accuracy. The accuracy results show that the DTs outperform the %-*based accuracy* when only the most frequent class is continually predicted. We also note a balanced apparition of the DT models.

Figure 5 illustrates the difference between the average DT sizes constructed by using our *GA* schema and the Standard DTs sizes. Clearly,

Table 4: Experimental data

Dataset	*N°. Classes*	*N°. Attributes*	*N°. Instances*	*% Base Error*
Abalone	29	9	4177	83.50
Breast-cancer	2	10	286	29.72
Breast-w	2	10	699	34.47
Dermatology	6	35	366	69.39
Diabetes	2	9	768	34.89
Ecoli	8	8	336	57.44
Hepatitis	2	20	155	20.64
Parkinsons	2	23	195	24.61
Primary-tumor	22	17	339	75.22
Sonar	2	61	208	46.63
Soybean	19	36	683	86.65
Spam	2	58	4601	39.40
Waveform-5000	3	40	5000	66.16

Table 5: GA results using $conf_1$

Dataset	DT model	$\|A\|$	*DT size*	*AC*	F-measure
Abalone	SimpleCart	6	985	26.14	0.14
Breast-cancer	BFTree	6	31	78.57	0.64
Breast-w	REPTree	6	17	100.0	1.00
Dermatology	SimpleCart	20	17	97.22	0.95
Diabetes	BFTree	7	91	81.58	0.77
Ecoli	REPTree	6	43	87.88	0.55
Hepatitis	SimpleCart	10	29	93.33	0.82
Parkinsons	REPTree	11	11	100.0	1.00
Primary-tumor	SimpleCart	13	61	51.52	0.19
Sonar	REPTree	17	32	80.00	0,78
Soybean	BFTree	22	157	97.06	0.91
Spam	BFTree	33	187	91.96	0.92
Waveform-5000	REPTree	23	401	77.20	0.77

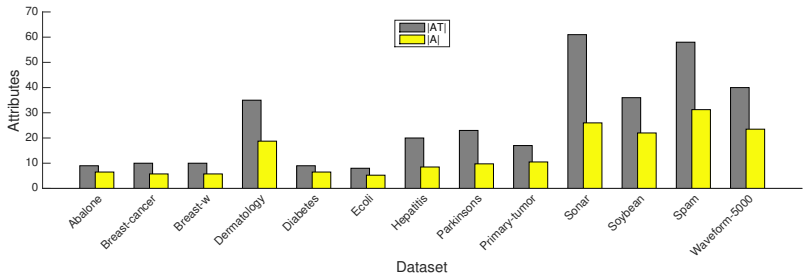

Figure 6: Dataset attributes $|AT|$ sizes vs. GA attributes used average sizes $|\overline{A}|$

in except diabetes dataset where the DT constructed using the GA schema is larger the DT constructed following a standard process, the GA results are less complicated than the Standard ones in term of DTs sizes.

Figure 6 illustrates the differences between the average number of attributes that are used for the different GA configurations and the number of attributes in the dataset. The used attributes during the GA DT construction process are 42.95±14.66 less than the whole datasets attributes.

Table 6: GA results using $conf_2$

Dataset	DT model	$\|A\|$	*DT size*	*AC*	F-measure
Abalone	J48	6	1271	24.22	0.12
Breast-cancer	BFTree	5	25	82.14	0.67
Breast-w	REPTree	5	23	98.55	0.98
Dermatology	J48	19	25	100.0	1.00
Diabetes	REPTree	7	87	76.32	0.70
Ecoli	SimpleCart	6	19	90.91	0.55
Hepatitis	SimpleCart	9	17	93.33	0.82
Parkinsons	REPTree	9	11	100.0	1.00
Primary-tumor	SimpleCart	10	49	48.48	0.20
Sonar	REPTree	25	25	80.00	0.78
Soybean	J48	24	81	95.59	0.87
Spam	BFTree	26	159	93.91	0.94
Waveform-5000	BFTree	24	309	75.60	0.78

Tables 9 and 10 show WEKA's DTs that are built with 10-CV using the pruning phase (DTP), and by deactivating pruning (UDTP). Table 9 results show a difference from one model to another. Generally, the most larger DTs are built using the J48 model, and the smaller ones are built using REPTree and SimpleCart models. This phenomenon is due to the applied pruning technique. Reduced-error pruning (REPTree) and Error Complexity Pruning (SimpleCart) techniques prune a set of subtrees which show a rate of failure. EBP technique (J48) allows grafting for each selected subtree to be pruned on one of its subtrees if this shows a lower failure rate. Besides, pre-pruning results (BFTree(PrP)) showed a loss of performance compared to the post-pruning ones (BFTree(PsP)). Moreover, the DTs sizes are surprisingly much smaller. BFTree(PrP) results illustrate a snapshot of the over-pruning problem especially over Breast-cancer, Hepatitis, Primary-tumor, and Sonar datasets.

Table 10 shows the experimental results of WEKA's DTs that are built without using the pruning phase (UDT). The results of our approach outperform those of DTP and UDT approaches regarding accuracy. The attributes

Table 7: GA results using $conf_3$

Dataset	DT model	$\|A\|$	*DT size*	*AC*	F-measure
Abalone	REPTree	7	1288	22.30	0.14
Breast-cancer	SimpleCart	6	13	82.14	0.72
Breast-w	J48	7	11	98.55	0.98
Dermatology	SimpleCart	19	19	97.22	0.97
Diabetes	REPTree	7	107	80.26	0.74
Ecoli	SimpleCart	4	35	90.91	0.56
Hepatitis	SimpleCart	6	11	86.67	0.83
Parkinsons	SimpleCart	11	17	94.74	0.88
Primary-tumor	BFTree	9	67	54.54	0.20
Sonar	REPTree	29	23	90.00	0.89
Soybean	BFTree	19	103	95.59	0.85
Spam	BFTree	34	199	93.48	0.93
Waveform-5000	REPTree	25	391	77.80	0.78

and the training samples that are used inthe GA process are far smaller than the ones used to build DTP and UDT models. The sizes of the generated DTs are smaller than those of UDT, and larger than those of DTP. Furthermore, the approach handles the over-pruning on tested datasets.

Figure 7 plots a comparison between the average *AC* results of the GA approach UDT and DTP.

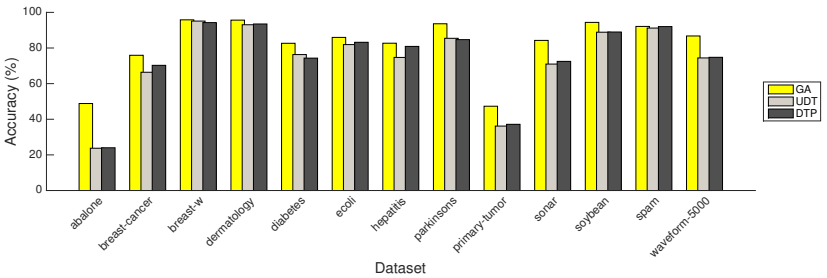

Figure 7: The average DTs AC performances

Table 8: GA results using $conf_4$

Dataset	DT model	$\|A\|$	*DT size*	*AC*	F-measure
Abalone	BFTree	7	1245	23.98	0.12
Breast-cancer	SimpleCart	6	21	89.29	0.56
Breast-w	J48	5	25	98.55	0.98
Dermatology	BFTree	17	15	100.0	1.00
Diabetes	REPTree	5	83	80.26	0.76
Ecoli	BFTree	5	11	87.88	0.49
Hepatitis	J48	9	11	93.33	0.82
Parkinsons	J48	8	17	100.0	1.00
Primary-tumor	SimpleCart	10	83	39.39	0.18
Sonar	REPTree	33	21	85.00	0.84
Soybean	SimpleCart	23	89	92.64	0.85
Spam	BFTree	32	238	93.91	0.93
Waveform-5000	REPTree	22	429	78.60	0.78

Table 9: DTP results using cross-validation technique

Dataset	BFTree(PrP)		BFTree(PsP)		J48		REPTree		SimpleCart	
	AC	*size*	*AC*	*size*	*AC*	*size*	*AC*	*size*	*AC*	*size*
Abalone	22.88	7	25.04	571	21.16	2312	24.89	256	26.07	21
Breast-cancer	68.18	1	67.83	5	75.52	6	70.62	32	69.23	1
Breast-w	93.84	7	94.27	25	94.56	27	93.84	17	94.84	15
Dermatology	93.98	19	93.98	19	93.98	40	91.53	29	93.98	17
Diabetes	73.82	5	73.56	5	73.82	39	75.26	49	75.13	5
Ecoli	82.14	13	84.22	35	84.22	43	81.54	21	83.92	29
Hepatitis	79.35	3	83.87	37	83.87	21	78.70	15	78.70	13
Parkinsons	84.10	7	87.69	19	80.51	23	85.64	11	85.64	13
Primary-tumor	26.25	1	39.82	73	39.82	88	38.93	46	41.00	17
Sonar	73.07	3	71.63	19	71.15	35	75.48	19	71.15	19
Soybean	86.23	75	91.36	145	91.50	93	84.77	89	91.06	129
Spam	89.13	31	92.74	195	92.97	207	92.89	95	92.43	149
Waveform-5000	69.64	21	75.60	277	75.08	659	76.90	181	76.68	129

Table 10: UDTP results using cross-validation technique

Dataset	BFTree		J48		REPTree		SimpleCart	
	AC	*size*	*AC*	*size*	*AC*	*size*	*AC*	*size*
Abalone	21.54	2131	20.58	2540	20.90	2327	21.35	2147
Breast-cancer	60.48	69	69.58	179	66.78	212	60.48	69
Breast-w	94.27	55	93.70	45	94.27	55	94.27	55
Dermatology	94.26	19	94.53	44	90.16	61	94.26	19
Diabetes	71.74	157	72.65	43	70.31	187	71.74	159
Ecoli	83.92	37	83.63	51	82.73	41	83.92	37
Hepatitis	80.00	49	80.64	31	78.06	23	80.00	49
Parkinsons	87.17	19	80.51	23	84.10	23	87.17	19
Primary-tumor	39.23	155	40.41	123	35.39	132	39.23	157
Sonar	70.67	27	69.71	35	73.07	35	70.67	27
Soybean	91.80	167	91.36	175	89.60	137	91.80	167
Spam	92.76	227	92.58	379	92.45	291	92.76	227
Waveform-5000	75.22	531	74.94	677	75.16	685	75.40	531

4.2. Comparison with existing state-of-the-art classifiers

To further evaluate the robustness of the proposed GA schema for DT construction, we compare our results with two state-of-the-art classifiers namely Naive Bayes (NB), and Support Vector Machine (SVM). For the two classifiers, we use WEKA's implementation with the default parameters. For SVM, we use two configurations. The first one is based on a Polynomial kernel denoted SVM(POLY) and the second one is based on a Radial Basis Function kernel denoted SVM(RBF).

Table 11 shows the obtained accuracy results, and it clearly demonstrates the robustness of the proposed approach compared to well established classifiers. As illustrated on Figure 8, our approach outperforms the other ones in eight datasets out of thirteen. However, NB, $SVM(POLY)$ and $SVM(RBF)$ scored best respectively for one, three, and one dataset out of the thirteen.

Table 12 presents the F-measure evaluation results for each classifier. These results support the domination of the proposed approach where the

Table 11: GA DTs accuracy vs. NB and SVM classifiers

Dataset	NB	SVM(POLY)	SVM(RBF)	GA
Abalone	22.45	25.64	19.40	24.34
Breast-cancer	76.43	67.50	60.00	78.57
Breast-w	96.23	96.67	93.77	98.91
Dermatology	95.83	95.57	94.17	98.06
Diabetes	75.00	76.58	76.05	77.10
Ecoli	83.33	84.24	87.27	84.85
Hepatitis	81.33	82.00	86.67	93.33
Parkinsons	69.47	89.47	93.16	93.68
Primary-tumor	45.46	42.73	41.21	44.85
Sonar	65.50	74.50	77.00	79.50
Soybean	93.24	93.38	75.41	93.38
Spam	80.02	90.24	85.54	92.89
Waveform-5000	79.30	86.92	97.38	76.56

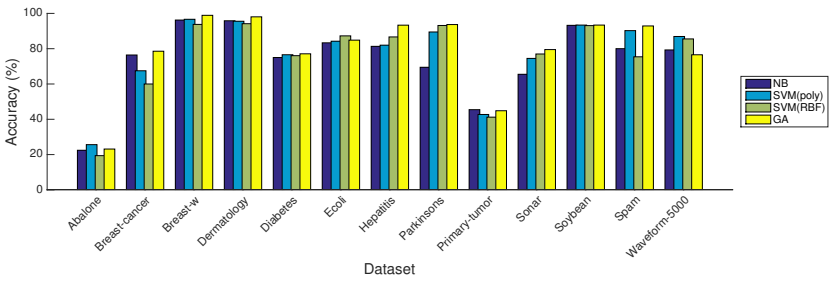

Figure 8: The GA DTs AC performances vs. NB, $SVM(POLY)$ and $SVM(RBF)$

latter outperforms its competitor approaches in seven datasets. However, NB, $SVM(POLY)$ and $SVM(RBF)$ scored best each in two datasets respectively.

4.3. Comparison with other evolutionary DT approaches

In this section, we compare our GA scheme for constructing an improved DT to Ant-Tree-Miner [39] that uses an ant colony optimization algorithm for inducing an optimized DT. In [39], the authors showed that Ant-Tree-Miner outperformed multiple decision tree construction approaches including

Table 12: GA DTs F-measure vs. NB and SVM classifiers

Dataset	NB	SVM (poly)	SVM (RBF)	GA
Abalone	0.086	0.062	0.026	0.127
Breast-cancer	0.681	0.541	0.550	0.546
Breast-w	0.958	0.964	0.932	0.987
Dermatology	0.903	0.951	0.910	0.977
Diabetes	0.718	0.709	0.721	0.709
Ecoli	0.494	0.484	0.542	0.517
Hepatitis	0.703	0.638	0.730	0.818
Parkinsons	0.662	0.819	0.907	0.919
Primary-tumor	0.182	0.165	0.177	0.167
Sonar	0.640	0.726	0.758	0.771
Soybean	0.821	0.870	0.842	0.841
Spam	0.794	0.896	0.692	0.924
Waveform-5000	0.780	0.870	0.854	0.763

state-of-the-art algorithms like C4.5 and CART as well as other evolutionary decision tree construction approaches like cACDT [7]. The comparison between the two algorithms will focus on accuracy and runtime. Table 13 presents the comparison results. We note that Ant-Tree-Miner required high training runtime for some datasets and it did not finish within 24 hours. Thus, we denote the accuracy and runtime by "- - -" in those cases.

The comparison in Table 13 clearly indicates that the DTs generated by our approach are more accurate than those generated by Ant-Tree-Miner, except in the case of Ecoli dataset. According to the runtime results, the proposed GA still outperforms the Ant-Tree-Miner. The results show that Ant-Tree-Miner could not finish within 24h for Abalone, Soybean, Spam and Waveform-5000 datasets. Moreover, Ant-Tree-Miner required considerably more running time to construct its DTs for six datasets from the nine for which it finished running within 24h.

4.4. Fault diagnosis in a rotating machine experimental results

We now consider the generation of DTs using our approach for the problem of fault diagnosis in rotating machines. The condition-monitoring task may be naturally treated like a classification task, where each condition (good

Table 13: Accuracy and runtime of *GA* DTs vs. Ant-Tree-Miner

Dataset	GA		Ant-Tree-Miner	
	AC	Runtime	AC	Runtime
Abalone	23.14	4702 s	- - -	- - -
Breast-cancer	78.57	373 s	72.70	482 s
Breast-w	98.91	230 s	95.42	1126 s
Dermatology	98.06	2019 s	94.25	945 s
Diabetes	77.10	324 s	73.57	37668 s
Ecoli	84.85	196 s	85.80	1028 s
Hepatitis	93.33	342 s	83.12	244 s
Parkinsons	93.68	377 s	90.79	345 s
Primary-tumor	44.85	375 s	44.26	11265 s
Sonar	79.50	1033 s	66.81	1130 s
Soybean	93.38	3212 s	- - -	- - -
Spam	92.89	10354 s	- - -	- - -
Waveform-5000	76.56	7501 s	- - -	- - -

and defective) is considered as a class. This is performed by extracting information from vibration sensors to indicate the machine's current condition (class). The proposed evolutionary approach is validated on a dataset that was presented in [29]. This dataset is constructed using a test ring (noted ring_data) under a normal operating condition and with three different faults: mass imbalance, gear fault, and faulty belt. The dataset is characterized as follow:

- ring_data: 4 classes, 55 attributes, 420 instances, 105 instances for each class, and 75% Base Error.

Table 14 shows the classification results of the proposed approach over the ring data. In general, the results are good where the *AC* illustrates performances between 97.62 and 100%, and the F-measure performances are between 0.97 and 1.00. The sizes of the DTs are the between 21 and 25 nodes. Yet, the used subset of attributes shows an important reduction compared to the total number of attributes which is 55.

Tables 15 and 16 show the experimental results with 10-CV by considering the pruned/unpruned WEKA's DTs.

Table 14: GA results on fault diagnosis

Dataset	DT model	$\|A\|$	*DT size*	*AC*	F-measure
$conf_1$	J48	27	23	100.0	1.00
$conf_2$	SimpleCart	27	23	97.62	0.97
$conf_3$	REPTree	43	25	97.62	0.97
$conf_4$	J48	26	21	100.0	1.00

Table 15: Standard DTs accuracy and size results on fault diagnosis

	BFTree(PrP)		BFTree(PsP)		BFTree(UDT)		J48		REPTree		SimpleCart	
Validation	*AC*	*size*	*AC*	*size*	*AC*	*size*	*AC*	*size*	*AC*	*size*	*AC*	*size*
UDT 10-CV	-	-	-	-	90	35	92,14	33	92,38	43	90,23	35
DTP 10-CV	81,9	29	89,04	35	-	-	92,14	33	92,38	43	88,8	35

Table 16: Standard DTs F-measure results on fault diagnosis

Validation	BFTree(UDT)	BFTree(PrP)	BFTree(PsP)	J48	REPTree	SimpleCart
UDT 10-CV	0.90	-	-	0.92	0.92	0.90
DTP 10-CV	-	0.82	0.89	0.92	0.88	0.89

These results show low performances compared to the ones obtained using *GA* in Table 14. The best obtained *AC* and F-measure performances are 92.14% and 0.92 respectively when considering DTP and UDT construction scheme. These rates indicate less performance than the minimal *AC* and F-measure performances that are produced by the proposed GA. The sizes of DTs generated by GA are between 21 and 25 nodes which are in general smaller than the ones that are obtained when considering DTP or UDT.

To further demonstrate the relevance of the performances of the proposed *GA* scheme, we compare them to the those of Naive Bayes, SVM(POLY), SVM(RBF), and Ant-Tree-Miner on the fault diagnosis case. Table 17 presents the accuracy and F-measure results for the different classifiers.

Table 17: Accuracy and F-measure of GA DTs vs. NB, SVM and Ant-Tree-Miner classifiers on fault diagnosis

	NB		SVM(POLY)		SVM(RBF)		Ant-Tree-Miner		GA	
	AC	F_1	AC	F_1	AC	F_1	AC	F_1	AC	F_1
10-CV	72.38	0.71	86.19	0.86	97.38	0.97	96.03	0.96	96.67	0.96

The obtained results show a low performance in terms of accuracy and F-measure for NB and $SVM(POLY)$ compared to the other classifiers. Although $SVM(RBF)$ scored best, we notice that the performances of $SVM(RBF)$, Ant-Tree-Miner and GA are very close.

Table 18: Wilcoxon Signed-Rank Test of GA vs. NB, SVM, DTP, UDT and Ant-Tree-Miner

	N	pos ranks	neg ranks	p-value	W-value	W critical value for N	Z-value
NB	14	11	94	0.00906	11	21	-2.6052
SVM(POLY)	14	13	78	0.02320	13	21	-2.2713
SVM(RBF)	14	20	85	0.04136	20	21	-2.0402
DTP	14	6	99	0.00350	6	21	-2.9191
UDT	14	0	105	0.00096	0	21	-3.2958
Ant-Tree-Miner	10	3	52	0.01242	3	8	-2.4973

Before concluding, we perform a Wilcoxon Signed-Rank Test [47] to determine if the new method accuracy performances (noted B observations, in short B_{obs}) tend to give higher scores than NB, SVM(POLY), SVM(RBF), the traditional decision tree constructing methods (*i.e.* DTP, UDT) and Ant-Tree-Miner (noted A observations, in short B_{obs})?. This test is conducted using the accuracy results of the proposed GA approach against the accuracies of NB, SVM(POLY), SVM(RBF), Ant-Tree-Miner, DTP and UDT, where we look to confirm the hypothesis :$A_{obs} < B_{obs}$. We note that for DTP and UDT comparison, we consider the best accuracy performance on each dataset (*e.g.*, in ring data case, 92.28 is the best AC out of the AC performances of BFTree, J48, REPTree and SimpleCart when pruning is used). Based on the 14 datasets the results in Table 18 show that the test is significant at $p \leq 0.05$ which allow us to reject the null hypothesis. The Z-value results are greater than the critical threshold at 1.96. Thus, we can conclude that our GA approach has significantly different results compared to NB,

SVM(POLY), SVM(RBF), DTP, UDT and Ant-Tree-Miner. Therefore, the *GA* approach is the best under the confidence of 95%.

5. Conclusion and further study

The popularity of DTs is strongly related to their simplicity, ease of understanding, and closeness to human reasoning. However, each DT model has its specificities, and many choices still need to be made making it a combinatorial problem. In this paper, we proposed to use good sub-training and sub-testing samples and only a subset of pertinent attributes to construct an optimal DT with respect to the input dataset. We also considered the use of the prune/unpruned decision as well as some other parameters when constructing the optimal DT. We proposed the usage of a GA scheme in order to overcome the high runtime and computational cost that are due to the combinatorial nature of the problem of choosing the best settings. Experimental results on 13 benchmark datasets showed that the proposed approach allows constructing efficient DTs that offer very high accuracy and F-measure results while efficiently reducing the size of the generated trees. Moreover, the proposed GA scheme was applied on a real-world application of fault diagnosis in rotating machines. The obtained results also demonstrated the effectiveness and efficiency of our approach in terms of complexity and classification accuracy of the constructed DTs. One of the main drawbacks of this work is that the encoding of the dataset attributes is in the form of a binary sequence where each bit represents the presence or the absence of the attribute in the GA candidate solution. In cases were the number of attributes is extremely large, the GA could suffer computational problems mainly due to the size of the binary encoding vector. An important future extension could be to propose an alternative encoding representation vector that requires a small size and that could efficiently handle large dimensions in a compressed way in order to overcome memory and computational limitations.

References

[1] S. Aksoy, K. Koperski, C. Tusk, G. Marchisio, and J.C. Tilton. Learning bayesian classifiers for scene classification with a visual grammar. *Geoscience and Remote Sensing, IEEE Transactions on*, 43(3):581–589, 2005.

[2] S. Aridhi, H. Sghaier, M. Zoghlami, M. Maddouri, and E.M. Nguifo. Prediction of ionizing radiation resistance in bacteria using a multiple instance learning model. *Journal of Computational Biology*, 23(1):10–20, 2016.

[3] R.C. Barros, M.P. Basgalupp, A.A. Freitas, and A.C.P.L.F. De Carvalho. Evolutionary design of decision-tree algorithms tailored to microarray gene expression data sets. *IEEE Transactions on Evolutionary Computation*, 18(6):873–892, 2014.

[4] P. Bermejo, L. de la Ossa, J.A. Gámez, and J.M. Puerta. Fast wrapper feature subset selection in high-dimensional datasets by means of filter re-ranking. *Knowledge-Based Systems*, 25(1):35–44, 2012.

[5] C. Blake and C.J. Merz. Uci repository of machine learning databases [http://www. ics. uci. edu/˜ mlearn/mlrepository. html]. irvine, ca: University of california. *Department of Information and computer science*, 55, 1998.

[6] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109 – 132, 2013.

[7] U. Boryczka and J. Kozak. An adaptive discretization in the acdt algorithm for continuous attributes. In *International Conference on Computational Collective Intelligence*, pages 475–484. Springer, 2011.

[8] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[9] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and regression trees*. CRC press, 1984.

[10] S.H. Cha and C. Tappert. A genetic algorithm for constructing compact binary decision trees. *Journal of pattern recognition research*, 4(1):1–13, 2009.

[11] B. Chandra and P.P. Varghese. Moving towards efficient decision tree construction. *Information Sciences*, 179(8):1059–1069, 2009.

[12] J. Chen, X. Wang, and J. Zhai. Pruning decision tree using genetic algorithms. In *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on*, volume 3, pages 244–248. IEEE, 2009.

[13] K.M. Chung, W.C. Kao, C.L. Sun, L.L Wang, and C.J Lin. Radius margin bounds for support vector machines with the rbf kernel. *Neural computation*, 15(11):2643–2681, 2003.

[14] R. Conforti, M. de Leoni, M. La Rosa, W.M.P. van der Aalst, and A.H.M. ter Hofstede. A recommendation system for predicting risks across multiple business process instances. *Decision Support Systems*, 69:1–19, 2015.

[15] UniProt Consortium et al. Uniprot: a hub for protein information. *Nucleic Acids Research*, 43(D1):D204, 2014.

[16] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[17] E. De la Hoz, E. de la Hoz, A. Ortiz, J. Ortega, and A. Martínez-Álvarez. Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps. *Knowledge-Based Systems*, 71:322–338, 2014.

[18] W. Dhifli and A.B. Diallo. Protnn: fast and accurate protein 3d-structure classification in structural and topological space. *BioData Mining*, 9(1):30, 2016.

[19] W. Dhifli, R. Saidi, and E.M. Nguifo. Smoothing 3d protein structure motifs through graph mining and amino acid similarities. *Journal of Computational Biology*, 21(2):162–172, 2014.

[20] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000.

[21] F. Esposito, D. Malerba, G. Semeraro, and J.A. Kay. A comparative analysis of methods for pruning decision trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(5):476–491, 1997.

[22] J. Gehrke, V. Ganti, R. Ramakrishnan, and W.Y. Loh. Boatoptimistic decision tree construction. In *ACM SIGMOD Record*, volume 28, pages 169–180. ACM, 1999.

[23] A.E. Gutierrez-Rodríguez, J.F Martínez-Trinidad, M. García-Borroto, and J.A. Carrasco-Ochoa. Mining patterns for clustering on numerical datasets using unsupervised decision trees. *Knowledge-Based Systems*, 82:70–79, 2015.

[24] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[26] S. Haykin and N. Network. A comprehensive foundation. *Neural Networks*, 2(2004), 2004.

[27] T.P. Hong, Y.L. Liou, S.L. Wang, and B. Vo. Feature selection and replacement by clustering attributes. *Vietnam Journal of Computer Science*, 1(1):47–55, 2014.

[28] N.E.I. Karabadji, I. Khelf, H. Seridi, and L. Laouar. Genetic optimization of decision tree choice for fault diagnosis in an industrial ventilator. In *Condition monitoring of machinery in non-stationary operations*, pages 277–283. Springer, 2012.

[29] N.E.I. Karabadji, H. Seridi, I. Khelf, N. Azizi, and R. Boulkroune. Improved decision tree construction based on attribute selection and data sampling for fault diagnosis in rotating machines. *Engineering Applications of Artificial Intelligence*, 35:71–83, 2014.

[30] N.E.I. Karabadji, H. Seridi, I. Khelf, and L. Laouar. Decision tree selection in an industrial machine fault diagnostics. In *Model and Data Engineering*, pages 129–140. Springer, 2012.

[31] K.M. Kim, J.J. Park, M.H. Song, I.C. Kim, and C.Y. Suen. Binary decision tree using genetic algorithm for recognizing defect patterns of cold mill strip. In *Advances in Artificial Intelligence*, pages 461–466. Springer, 2004.

[32] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[33] D.D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning*, pages 4–15. Springer, 1998.

[34] H. Lu, R. Setiono, and H. Liu. Effective data mining using neural networks. *Knowledge and Data Engineering, IEEE Transactions on*, 8(6):957–961, 1996.

[35] M. Macaš, L. Lhotská, E. Bakstein, D. Novák, J. Wild, T. Sieger, P. Vostatek, and R. Jech. Wrapper feature selection for small sample size data driven by complete error estimates. *Computer methods and programs in biomedicine*, 108(1):138–150, 2012.

[36] J. Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine learning*, 4(2):227–243, 1989.

[37] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C.A.C. Coello. A survey of multiobjective evolutionary algorithms for data mining: Part i. *IEEE Transactions on Evolutionary Computation*, 18(1):4–19, 2014.

[38] T. Niblett and I. Bratko. Learning decision rules in noisy domains. In *Proceedings of Expert Systems' 86, The 6Th Annual Technical Conference on Research and development in expert systems III*, pages 25–34. Cambridge University Press, 1987.

[39] F.E.B. Otero, A.A. Freitas, and C.G. Johnson. Inducing decision trees with an ant colony optimization algorithm. *Applied Soft Computing*, 12(11):3615–3626, 2012.

[40] S. Pei, Q. Hu, and C. Chen. Multivariate decision trees with monotonicity constraints. *Knowledge-Based Systems*, 112:14–25, 2016.

[41] S. Piramuthu. Input data for decision trees. *Expert Systems with applications*, 34(2):1220–1226, 2008.

[42] J.R. Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[43] J.R. Quinlan. C4. 5: Programming for machine learning. *Morgan Kauffmann*, page 38, 1993.

[44] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[45] M. Srinivas and L.M. Patnaik. Genetic algorithms: A survey. *Computer*, 27(6):17–26, 1994.

[46] G. Stein, B. Chen, A.S. Wu, and K.A. Hua. Decision tree classifier for network intrusion detection with ga-based feature selection. In *Proceedings of the 43rd annual Southeast regional conference-Volume 2*, pages 136–141. ACM, 2005.

[47] F. Wilcoxon, S.K. Katti, and R.A. Wilcox. Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test. *Selected tables in mathematical statistics*, 1:171–259, 1970.

[48] Q. Wu, Y. Ye, H. Zhang, M.K Ng, and S.S. Ho. Forestexter: an efficient random forest algorithm for imbalanced text categorization. *Knowledge-Based Systems*, 67:105–116, 2014.

[49] S. Xiang, F. Nie, G. Meng, C. Pan, and C. Zhang. Discriminative least squares regression for multiclass classification and feature selection. *IEEE transactions on neural networks and learning systems*, 23(11):1738–1754, 2012.